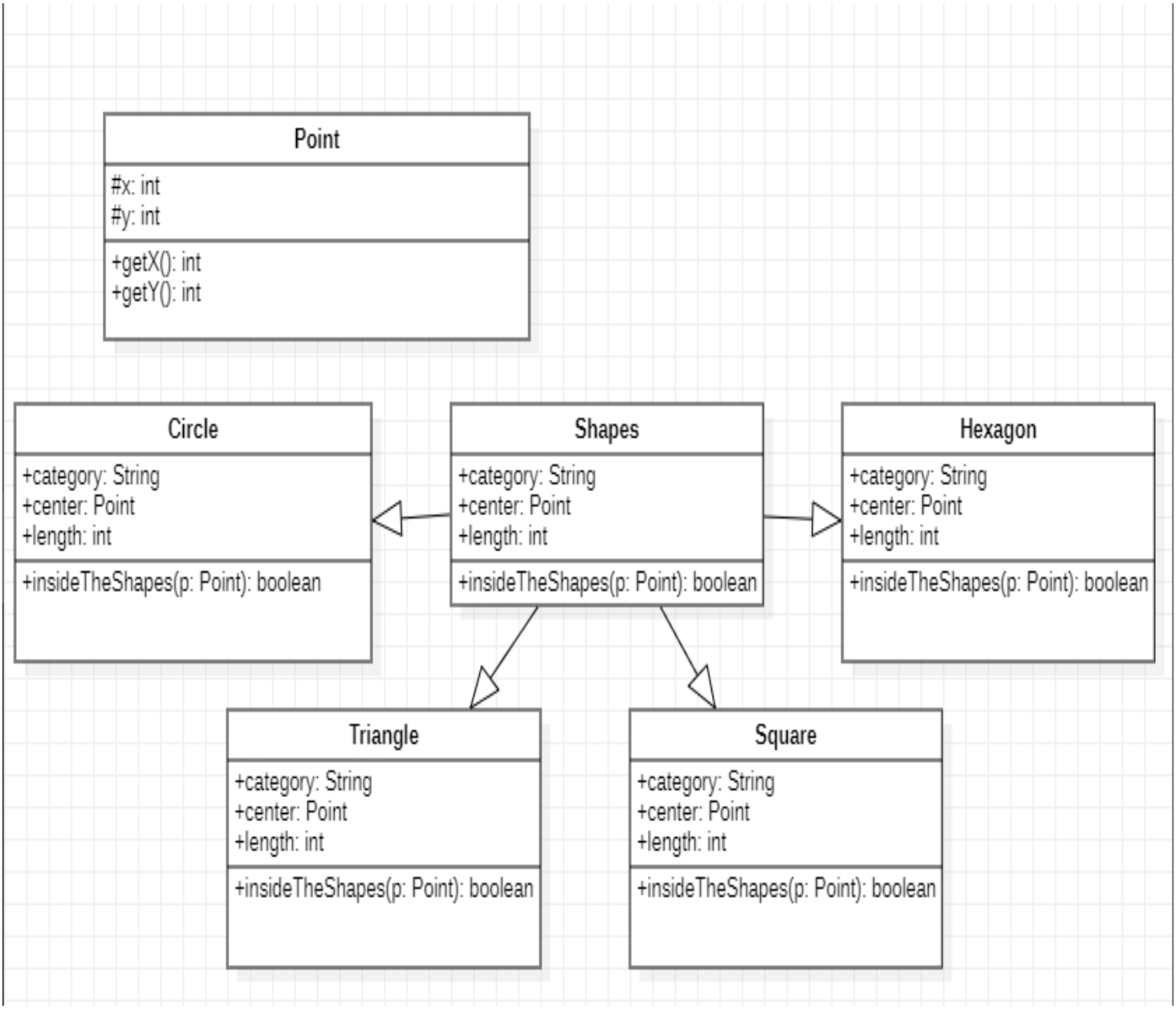
The documentation should contain:

- **the description of the exercise,**

Choose a point on the plane, and fill a collection with several regular shapes (circle, regular triangle, square, regular hexagon). How many shapes contain the given point?

Each shape can be represented by its center and side length (or radius), if we assume that one side of the polygons are parallel with x axis, and its nodes lies on or above this side. Load and create the shapes from a text file. The first line of the file contains the number of the shapes, and each following line contain a shape. The first character will identify the type of the shape, which is followed by the center coordinate and the side length or radius. Manage the shapes uniformly, so derive them from the same super class.

- **the class diagram**,



- **the short description of each methods,**

Point class:

getX(): int-> returns x, protected attribute of class Point;

getY(): int-> returns y, protected attribute of class Point;

Shapes class:

insideTheShapes(p: Point): boolean -> point(x, y) will be given from the console as a parameter and this function decides whether that point lays inside the Shapes(Circle, Triangle, Square, Hexagon) or not, if yes then returns true.

Database class:

read(fileName: String) -> reads the input file and creates Collection of objects(Shapes), throws exception in case of incorrect file;

report() -> prints each shape from the Collection;

numberOfShapes() -> prints out shapes that contains the point(x, y) for itself and total number.

How to run the program:

Terminal must be cmd for testing, not PowerShell!!!

```
D:\Java\assignment1>javac -cp .;junit-4.12.jar;hamcrest-core-1.3.jar ShapesTest.java

D:\Java\assignment1>java -cp .;junit-4.12.jar;hamcrest-core-1.3.jar org.junit.runner.JUnitCore ShapesTest
JUnit version 4.12
.....
Time: 0.012

OK (5 tests)


D:\Java\assignment1>javac Main.java

D:\Java\assignment1>java Main
Give the name of the file:
Enter the file: input.txt
Shapes in the file:
Category : C, center: x = 4, y =  6, length: 7
Category : H, center: x = 3, y =  4, length: 5
Category : S, center: x = 4, y =  8, length: 5
Category : T, center: x = 2, y =  3, length: 1
Enter the x:
2
Enter the y:
3
Category : C, center: x = 4, y =  6, length: 7
Category : H, center: x = 3, y =  4, length: 5
2 shapes contain the given point.
```

- **and the testing (white box / black box)**

WHITE BOX TESTING

1) Checkig attributes of Point class:

```
 Point p = new Point(3, 3);

       assertEquals(3, p.getX());
       assertEquals(3, p.getY());
```

2) Checking Triangle class:

```
 Triangle s = new Triangle(new  Point(2, 4), 8);

       assertNotEquals(0, s.length);
       assertEquals("T", s.category);
       assertTrue(s.insideTheShape(new Point(1, 5)));
       assertFalse(s.insideTheShape(new Point(7, 12)));
```

3) Checking Circle class:

```
Circle s = new Circle(new  Point(5, 3), 9);

       Circle c = new Circle(new  Point(2, 7), 8);
       assertNotEquals(new Point(8, 9), c.center);
       assertEquals(9, s.length);
       assertFalse(s.insideTheShape(new Point(13, 15)));
       assertTrue(s.insideTheShape(new Point(3, 5)));
```

4) Checking Square class:

```
Square s = new Square(new  Point(-2, -3), 9);

       Square c = new Square(new  Point(4, 0), 2);
       assertNotEquals(new Point(8, 9), c.center);
       assertEquals(9, s.length);
       assertFalse(s.insideTheShape(new Point(13, 15)));
       assertTrue(c.insideTheShape(new Point(3, 1)));
```

5) Checking Hexagon class:

```
Hexagon s = new Hexagon(new  Point(-5, 3), 9);

       Hexagon c = new Hexagon(new  Point(2, -7), 18);
       assertNotEquals(new Point(8, 9), c.center);
       assertEquals(9, s.length);
       assertFalse(s.insideTheShape(new Point(1, 15)));
       assertTrue(c.insideTheShape(new Point(3, -5)));
```

BLACK BOX TESTING

1) Test cases for file handling

***Not existing file** -> if you give output.txt(not existing file), it must give “File not found!” message;

***Empty file** -> if text file starts with letters(not number) or empty file, then it must give “Empty file or Invalid starting!” error message;

***Not matching character** -> if first character of every line after the number is not one of (“S”, “C”, “H”, “T”), then it must give an error message, saying “Invalid input!”;

***Negative length** -> if length of the any shape is negative, then program must catch the exception.

2) Test cases for Database class

***Format of the each shape in the database:**

Database data = new Database();

db.report() :

-firstly it must ask for the input file;

-if file finds successfully, then it must create array of objects

(“C” -> Circle, “S” -> Square, “T” -> Triangle, “H” -> Hexagon);

- if everything matches, console must print out in this format:

```
Enter the file: input.txt
Shapes in the file:
Category : C, center: x = 4, y =  6, length: 7
Category : H, center: x = 3, y =  4, length: 5
Category : S, center: x = 4, y =  8, length: 5
Category : T, center: x = 2, y =  3, length: 1
```

***Correct number of shapes where given Point lays:**

db.numberOfShapes();

-insideTheShapes(Point p) is correct, then +1.

UPOL SAIPIDIN KYZY(IHLG15)

16/10/2022